Send Orders for Reprints to reprints@benthamscience.ae

128 *The Open Cybernetics & Systemics Journal*, 2014, *8*, 128-138

Open Access

Double Layer Based Multi-label Classifier Chain

Tao Guo* and Guiyang Li

College of Computer Science, Sichuan Normal University, Chengdu, Sichuan, 610101, P.R. China

Abstract: In multi-label learning, each training example is associated with a set of labels and the task is to predict the proper label set for each unseen instance. The widely known binary relevance method (BR) for multi-label classification considers each label as an independent binary problem. It is ignored in the literature due to inadequacy of not considering label correlations. In this paper, we present our double layer based classifier chains method (DCC), which overcomes disadvantages of BR and inherits the benefit of classifier chain method (CC). This algorithm decomposes the multi-label classification problem into two classification processes to generate classifier chain. Each classifier in the chain is responsible for learning and predicting the binary association of the label given the attribute space expanded by all prior binary relevance predictions in the chain. This chaining allows DCC to take into account correlations in the label space. We also extend this approach further in an ensemble framework. An extensive evaluation covers a broad range of multi-label datasets with a variety of evaluation measures specifically designed for multi-label classification. Experiments on benchmark datasets validate the effectiveness of proposed approach comparing with state-of-art methods in terms of average ranking.

Keywords: Binary relevance, classifier chain, evaluation method, multi-label classification, multiple layers.

1. INTRODUCTION

Traditional single-label classification is concerned with learning from a set of examples that are associated with a single-label from a set of disjoint labels. Multi-label learning is concerned with learning from instances, where each instance is associated with multiple labels. The main task from multi-label learning is to predict the label sets of unseen instances through analyzing training examples with known label sets. These known sets belong to a predefined set of labels [1-3]. Each example in the training set is represented by a feature vector and associated with one set of labels. Generally, multi-label learning can be categorized into multi-label classification and multi-label ranking [3]. The goal of multi-label classification is to construct a classifier or predictive model to provide a list of relevant labels for unseen instance. However, the purpose of the multi-label ranking is to build a predictive model that provides a list of preferences of the labels from the set of possible labels [3].

Early research on learning from multi-label data focused on automated document categorization [4]. As the ability to collect and store large sets of data increased in recent years, multi-label learning has recently received significant attention from machine learning community [5]. It motivated an increasing number of new applications and involved a wide variety of domains, including text classification [6], scene and video classification [7], and bioinformatics [8]. The algorithms for multi-label classification can be categorized into two different groups [5]: problem transformation methods and algorithm adaptation methods.

Problem transformation methods transform the multi-label classification task into one or more single-label classification, regression or raking tasks [6]. Prior problem transformation approaches have employed algorithms such as Support Vector Machines, Naïve Bayes and K-Nearest Neighbor methods [7]. These approaches can be grouped into three categories: binary relevance, label power-set and pair-wise methods [8]. Algorithm adaptation methods extend specific learning algorithms in order to deal with multi-label data directly [6]. Some well-known approaches involve decision trees, AdaBoost [7]. This paper we focus on the problem transformation methods.

The most common and widely-used problem transformation method is the BR [8]. It learns one binary classifier for each different label, such that each binary classifier is trained to predict the relevance of one of the labels. For the classification of a new instance, BR outputs the union of the labels that are predicted by the classifiers positively. However, during the learning processing, BR simply learns independent binary classifiers for each label, and ignores the correlations existing between labels in the training data

Label power-set (LP) [9] multi-label learning method takes label correlations into consideration. It considers each unique set of labels that exits in a multi-label training set as one of the classes of a new single-label classification task. The output of LP for a new instance is the most probable class that is a set of label [10]. It can achieve better performance compared with BR [8]. However, LP is challenged by application domains with large number of labels and training examples [11]. It also suffers from the data deviation since only small number of training examples is contained [12].

To handle these problems, G. Tsoumakas proposed random *k*-labelsets (RAKEL) approach [8]. This method breaks

1874-110X/14 2014 Bentham Open

the original set of labels into a number of small random subsets which is named labelsets and employs LP to train a corresponding classifier. The labelsets can be either disjoint or overlapping depending on which of two strategies is used to construct them [8]. The k in RAKEL is a parameter that specifies the size of the subsets. RAKEL takes the label correlations into account and avoids the LP's problem effectively. However, in order to achieve optimal performance, some parameters, such as subset k, number of models, threshold, must be perform internal cross validation. It is a harsh work to find the optimal parameters [12].

Godbole [13] presents meta-MR algorithm (MBR) which extends the SVM binary classifiers with two stages. In this algorithm, the authors stacked BR classification outputs along with the full original feature space into a separate meta classifier to create a two-stage classification process. This process takes label correlations into consideration, but the meta classifier implies an extra iteration of both training and testing data as well as internal classifications on the training data to acquire the label outputs for this meta step [4, 14].

Proposed in [4], the classifier chains method has become one of the most popular methods for taking label correlations into account. CC offers a general problem transformation method that inherits the efficiency of BR and overcomes the disadvantages of BR. The CC consists of q training binary classifiers, one for each label. These classifiers are connected at random order in a chain, such that the classifier connected with the instance can be used as input features not only the instance, but also the output predictions of the previous classifiers.

Although CC computes with the high accuracy of more computationally complex methods, it still has a drawback in that the label ordering in CC is composed at random. Eduardo [15] demonstrated that the label ordering of CC has a strong effect on predictive accuracy. To deal with this problem, [15] proposed a genetic algorithm for optimizing the label ordering in classifier chains. Since some of the parameters in genetic algorithm are sensitive, the manually parameters setting may reduce the predictive performance. Also, in CC, the first binary classifiers will possibly generate wrong predictions and course significant error propagation along the chain [14, 15]. To solve this problem, the authors of CC propose combining random orders through an ensemble of classifier chains (ECC) in [4, 14]. ECC trains m CC classifiers. Each classifier is trained with a random chain ordering of L or a random subset of D and makes label predictions. The predictions are summed by label so that each label receives a number of votes. A threshold is used to select the most popular labels, which form the final predicted label set. The experimental results performed in [14] show that the ECC performed high prediction as compared with other multi-label methods. The notable potential issue of ECC is that a potentially very large number of instances must be processed when an ensemble of binary transformations. This process is timing consume and memory complexity [14]. The other disadvantages of the ECC approach lies in the fact that it cannot be applied when the goal is to build interpretable classifiers. These kinds of classifiers explain their classification decisions and are mainly represented by decision trees [16] and associative classifiers [17].

In this paper, we outline the advantages of BR-based methods and present our double layer based classifier chains method, which overcomes disadvantages of the binary method and inherits the benefit of classifier chain method. An ensemble framework for classifier chains called EDCC is also introduced. The performances of proposed methods are compared with familiar multi-label classification methods on a wide range of datasets under four state-of-art multi-label evaluation metrics.

The rest of paper is organized as follows. In section 2, we proposed the DCC algorithms in training and testing stages. The procedures for BR, CC, MBR, and DCC are also diagramed for further comparison. In section 3, we present the implementation of ensembles classifier chains (EDCC). Section 4 describes datasets, experimental setups and evaluation measures involved in the experiments. The results of comparative experiments and analysis are presented in section 5. Finally, we conclude the paper in section 6.

2. DOUBLE LAYER BASED CLASSIFIER CHIAN

2.1. Preliminaries

Let $X^d \subset \mathbb{R}$ be the input domain of all possible attribute values. An instance $\mathbf{x} \in X$ is represented as a vector of d attribute values $\mathbf{x} = [x_1, \cdots, x_d]$. The set $\mathbb{L} = \{1, \cdots, L\}$ is the output domain of possible labels. Each instance $\mathbf{x}$ is associated with a subset of these labels. This subset is represented by a L-vector $\mathbf{y} = [y_1, \cdots, y_L]$, where $y_j = 1$, if label j is associated with instance $\mathbf{x}$, and 0 otherwise. We also define a set of training data D of N labeled examples $D = \{(\mathbf{x}^i, \mathbf{y}^i) \mid i = 1, \cdots, N\}$. $\mathbf{h}^f = (h_1^f, \cdots, h_L^f)$ and $\mathbf{h}^s = (h_1^s, \cdots, h_L^s)$ are the classifiers constructed in the first layer and second layer respectively. The final output $\hat{y} \in \{0,1\}^L$ is induced for any instance $\mathbf{x}$.

2.2. The DCC Framework

The method sets two layers to decompose the multi-label classification problem into independent binary classification problems. In first layers, DCC model involves L binary transformations-one for each label and each binary model is trained to predict the relevance of one of the labels [4]. In second layer, the instance with attribute space for each binary model is extended with label relevance of all previous classifiers to form a classifier chain. The algorithm follows the advantage of CC in that higher predictive performance is achieved by passing label correlation information along a chain of classifiers. But it is different from CC in that the attribute space for each binary model in second layer is augmented with the 0/1 label prediction coming from first layer classifiers and all prior binary relevance prediction from second layer in which the classifier chain is built. Each classifier in the chain is responsible for learning and predicting the binary association of the label given the attribute space. The classification process is composed of training and testing stages. The algorithms for the two stages of DCC are defined in the followings.

2.2.1. Training Stage

In the training stage, the training data involving in the first layer for each classifier is defined in (1):

$$D_j^f = \{(\mathbf{x}_i, y_j^i) \mid 1 \le i \le d\} \quad (1)$$

For any multi-label training example $(\mathbf{x}^i, \mathbf{y}^i)$, some binary algorithm B is used to induce a binary classifier $h_j^f : \boldsymbol{X} \to \{0,1\}$, *i.e.* $h_j^f \leftarrow \mathsf{B}(D_j^f)$.

The training data involving in the second layer for each classifier h_j^s is defined in (2):

$$D_j^s = \{([\mathbf{x}_i, y_1^i, \cdots, y_{j-1}^i, y_{j+1}^i, \cdots, y_L^i], y_j^i) \mid 1 \le i \le d\} \quad (2)$$

In this stage, we denote the binary classifier $h_j^s : X \times \{0,1\}^{L-1} \to \{0,1\}$, *i.e.* $h_j^s \leftarrow \mathsf{B}(D_j^s)$.

The pseudo-code of DCC algorithm for a training dataset D is presented in Algorithm 1.

Algorithm 1. DCC's training phase for training set D and label set L of L labels.

TRAINING($D = \{(\mathbf{x}^i, \mathbf{y}^i) \mid i = 1, \cdots, N\}$)
1 for j = 1,…, L
2 Construct the first layer training set D_j^f according to Eq.(1);
3 $h_j^f \leftarrow \mathsf{B}(D_j^f)$
4 Endfor
5 for j= 1,…, L
6 Construct the second layer training set D_j^s according to Eq.(2);
7 $h_j^s \leftarrow \mathsf{B}(D_j^s)$
8 Endfor

After training, the classifiers $\mathbf{h}^f = (h_1^f, \cdots, h_L^f)$ and $\mathbf{h}^s = (h_1^s, \cdots, h_L^s)$ are constructed respectively.

2.2.2. Testing Stage

During the processing of testing stage, all instances with *d* attribute space are firstly classified by the classifier h_j^f in the first layer.

In second layer, the attribute space for each binary model is extended with the label relevance from first layer linking with the label relevances from all previous classifiers in second layer. The correlation for each label is fully considered after the procedure of the second layer. Also a chain $\mathbf{h}^s = (h_1^s, \cdots, h_L^s)$ of binary classifiers is formed to induce final output $\hat{\mathbf{y}} = [\hat{y}_1^s, \cdots, \hat{y}_L^s]$. The pseudo-code of DCC algorithm for the testing stage is presented in Algorithm 2.

The output of each classifier for unseen instance **x** in the first layer is shown in (3):

$$\hat{y}_j^f = h_j^f(\mathbf{x}) \quad (3)$$

In second layer, instance x combined with label relevances of all previous classifiers is used as input for each individual classifier. The output of each classifier is shown in (4):

$$\hat{y}_j^s = h_j^s([\mathbf{x}, \hat{y}_1^s, \cdots, \hat{y}_{j-1}^s, \hat{y}_{j+1}^f, \cdots, \hat{y}_L^f]) \quad (4)$$

The final output for instance x is presented in (5):

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}^s = [\hat{y}_1^s, \cdots, \hat{y}_L^s] \quad (5)$$

where, classifier **h** is consisted of L binary classifiers $h_1, ..., h_L$, and each h_j learns from D to predict the relevance of $\hat{y} \in \{0,1\}$. The pseudo-code of DCC algorithm for a test instance is presented in Algorithm 2.

Algorithm 2. DCC's prediction phase for a test instance x.

CLASSIFY(x)
1 ▷ global $\mathbf{h}^f = (h_1^f, \cdots, h_L^f)$, $\mathbf{h}^s = (h_1^s, \cdots, h_L^s)$
2 for j = 1,…, L
3 Generate the first layer output $\hat{y}_j^f$ according to Eq.(3);
4 Endfor
5 for j = 1,…, L
6 Generate the second layer output $\hat{y}_j^s$ according to Eq.(4);
7 Endfor
8 $\hat{\mathbf{y}} = [\hat{y}_1^s, \cdots, \hat{y}_L^s]$
9 return $\hat{\mathbf{y}}$

In this procedure, a binary classifier chain $\mathbf{h}^s = (h_1^s, \cdots, h_L^s)$ is formed. Each classifier h_i in the chain is responsible for learning and predicting the binary association of the *j*th label given the attribute space, augmented by all prior binary relevance predictions in the chain. The classification process begins at h_1 and propagates along the chain. This chaining method passes label information among classifiers, allowing DCC to take label correlations into consideration in the label space in the second layer, thus overcoming the label independence problem of BR.

2.3. Comparison of Related Binary Relevance Classifiers in the Procedure of Testing Stage

In order to demonstrate the procedure of DCC and understand the different algorithm ideas which are relate to binary relevance more clearly, the following section performs the comparison of the schemes for BR, CC, MBR, and DCC in testing stage. Figs. (**1-4**) show the procedure of transformations with an example under each method for (x, y), where unseen instance x contains three attribute spaces and the output for prediction $\hat{y} \in \{0,1\}$.

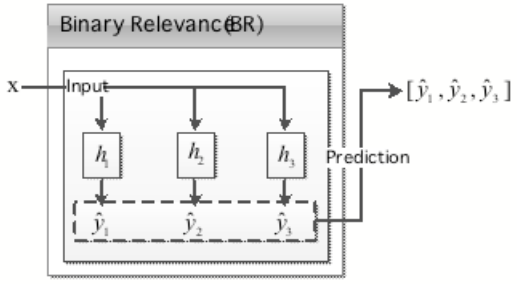

Fig. (1). The procedure of BR in testing stage.

(Fig. (**1**)) shows the procedure of BR. In Fig. (**1**), the multi-label learning problem is decomposed into three independent binary classification problems, where each binary classification problem corresponds to a possible label. The binary classifiers $[h_1, h_2, h_3]$ induced by some binary learning algorithm B in the training stage is utilized to perform prediction for unseen instance x and generate final prediction $[\hat{y}_1, \hat{y}_2, \hat{y}_3]$ without considering correlation among labels.

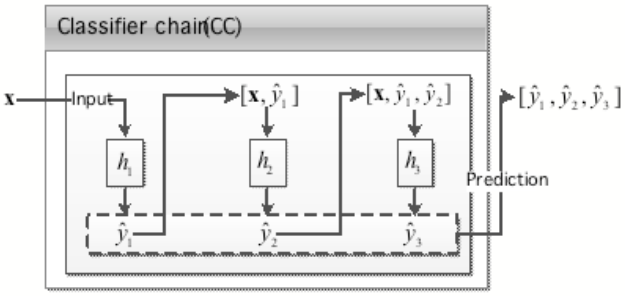

Fig. (2). The procedure of CC in testing stage.

(Fig. (**2**)) presents the scheme of CC. In Fig. (**2**), the unseen instance x is used as input to generate the first binary classifier h_1 under BR firstly. The corresponding binary testing set is constructed by appending each instance x with its relevance to the labels predicted by preceding classifier(s). Then, the subsequent binary classifier h_2 is built upon this binary testing set. Following this procedure, for any instance x, its associated label set $[\hat{y}_1, \hat{y}_2, \hat{y}_3]$ is predicted by traversing the classifier chain iteratively.

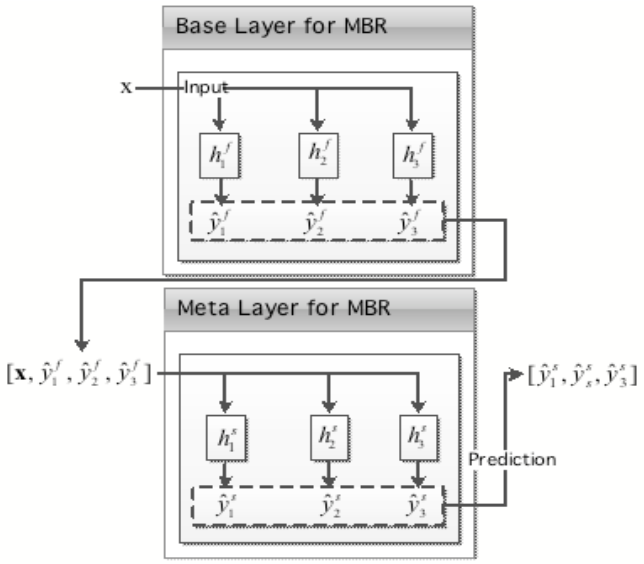

Fig. (3). The procedure of MBR in testing stage.

(Fig. (**3**)) shows the procedure of meta-BR (MBR) proposed by Godbole [13]. The method stacks BR classification outputs along with the full original feature space into a separate meta classifier to create a two-stage classification process. In base layer (or first layer), unseen instance **x** is the input for each of the three classifiers $[h_1^f, h_2^f, h_3^f]$, then pre-label prediction set $[\hat{y}_1^f, \hat{y}_2^f, \hat{y}_3^f]$ is obtained. In the meta layer (or second layer), the corresponding input dataset is constructed by appending each instance **x** with all labels predicted by preceding classifiers. The binary classifiers are also utilized to generate final prediction set $[\hat{y}_1^s, \hat{y}_2^s, \hat{y}_3^s]$. During this procedure, the label correlations are took into account.

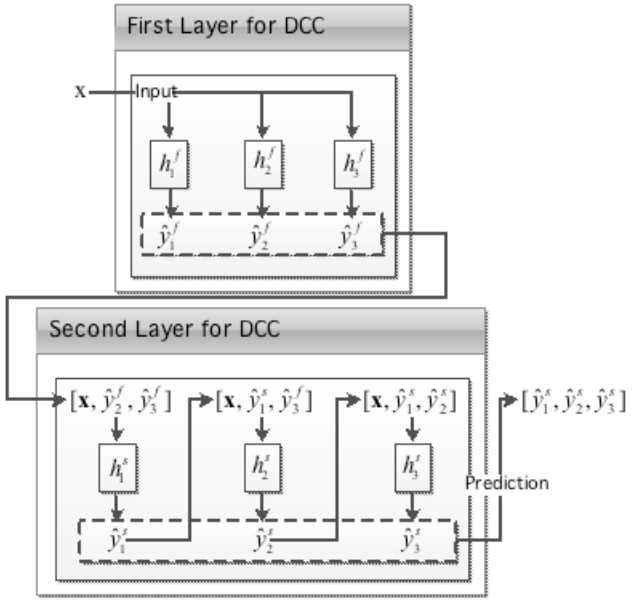

Fig. (4). The procedure of DCC testing stage.

(Fig. (**4**)) shows the procedure of proposed DCC method. From Fig. (**4**), we may see that the proposed method sets two layers to decompose the multi-label classification problem into independent binary classification problems. The procedure completed in the first layer is similar as BR. In this layer, the model involves *L* binary transformations-one for each label without considering label correlations and a pre-prediction label set $[\hat{y}_1^f, \hat{y}_2^f, \hat{y}_3^f]$ is induced. Same as CC, the unseen instance with attribute space for each binary model is extended with 0/1 label relevance of all previous classifiers to form a classifier chain $[h_1^s, h_2^s, h_3^s]$ in the second layer. Each classifier in the chain is responsible for learning and predicting the binary association of each label by traversing the classifier chain iteratively. Finally, the prediction set $[\hat{y}_1^s, \hat{y}_2^s, \hat{y}_3^s]$ is generated under this chain.

3. ENSEMBLES OF DCC

Since the label ordering in CC is composed at random [4, 14, 15], we design ensembles classifier chains under double layer to train *L* DCC classifiers $h_1, ..., h_L$. Each classifier is unique and able to give different multi-label predictions. These predictions are summed by label so that each label receives a number of votes. Pcard1 [14] is used as threshold to select the most popular label that form the final predicted label sets.

4. EXPERIMENTS

4.1. Data Sets

We try to include a considerable variety and scale of multi-label datasets, thus the proposed algorithm are evalu-

Table 1. A collection of multi-label datasets and associated statistics.

Dataset	*N*	*d*	*L*	LCard	Type
Music	592	71n	6	1.89	media
Genbase	661	1185b	27	1.25	biology
Medical	978	1449b	45	1.25	text
Langlog	1460	1004n	75	1.18	text
Scene	2407	294n	6	1.07	media
Yeast	2417	103n	14	4.24	biology
Enron	1702	1001b	53	3.38	text
Corel5K	5000	499b	374	3.52	image
Bibtex	7395	1836b	159	2.40	text

N indicates numeric attributes, *d* indicates binary attribute.

ated on 9 different benchmark datasets obtained from [18] with dimensions ranging from 6 to 374 labels. The main characteristics of each dataset are shown in (Table **1**). The values in the second column (*N*) represent the number of instances. The values in the third column (d) present the features. And the values in the fourth column (*L*) represent the label. The values in the fifth column (LCard) indicate the label cardinality, which is the average number of labels per instance [5, 14]. The formula for the label cardinality is defined for N examples as:

$$Lcard = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{\ell} y_j^i \tag{6}$$

The sixth column presents the application domain of each dataset. To evaluate the efficiency of DCC in different datasets, we refer to [14] to classify the datasets into three groups according to the complexity $(N \times d \times L)$ of each datasets: small ("Music", "Genbase", "Medical", "Medical", and "Langlog"), medium ("Scene", "Yeast", and "Enron") and large ("Corel5K", and "Bibtex").

4.2. Evaluation Methods

In multi-label experiment, it is essential to evaluate the performance of each algorithm. Over the last few years, several evaluation measures specifically designed for multi-label classification have been proposed in literatures, such as [1, 5, 14, 15, 19]. All of these evaluation methods help identifying how the proposed algorithms perform well crossing a range of evaluation methods. There is an important division between label-based evaluation, which is carried out on a per-label basis, and label set-based evaluation, which evaluates label sets. In label-based evaluation, the evaluation is executed on a per-label basis. On the other hand, the label set-based evaluation is carried out on label sets. In this experiment evaluation, we use several of both types of measures, including 0/1 loss, Hamming Loss, Precision, Accuracy, Micro_F1, and time for training and testing in this paper.

When any predicted set of labels $\hat{y}$ must match the true set of labels y exactly, this is known as *0/1 Loss* as a loss measure. It is defined in (7):

$$0/1\ \text{Loss} = 1 - \frac{1}{N}\sum_{i=1}^{N} 1_{y^i = \hat{y}^i} \tag{7}$$

Since 0/1 Loss can be seen very harsh, any label set not predicted perfectly is given a zero score [14]. However, when the label is a separate binary evaluation, the Hamming loss tends to be very tolerant due to the typical sparsity of multi-labelling, and ignores the multi-label problem as a whole. The *Hamming Loss* is defined in (8):

$$\text{Hamming Loss} = \sum_{i=1}^{N}\sum_{j=1}^{L} 1 y_j^i = \hat{y}_j^i \tag{8}$$

Accuracy measure introduced by Godbole in [15] is a label set-based measure. It is defined in (9):

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\frac{|\, y^i \cap \hat{y}^i \,|}{|\, y^i \cup \hat{y}^i \,|} \tag{9}$$

*Micro*_F1measure is an approach of combining the *Micro_precision* and *Micro_recall* measures of a classifier by means of an evenly harmonic mean of both them. It is defined in (10).

$$\text{Micro\_F1} = \frac{2 \times \text{Micro\_precision} \times \text{Micro\_recall}}{\text{Micro\_precision} \times \text{Micro\_recall}} \tag{10}$$

Where, the *Micro_precision* and *Micro_recall* measures are defined as (11) and (12) respectively.

$$\text{Micro\_precision} = \frac{\sum_{j=1}^{L} tp_j}{\sum_{j=1}^{L} tp_j + \sum_{j=1}^{L} fp_j} \tag{11}$$

$$\text{Micro\_recall} = \frac{\sum_{j=1}^{L} tp_j}{\sum_{j=1}^{L} tp_j + \sum_{j=1}^{L} fn_j} \tag{12}$$

where tp_j, fp_j and fn_j are the number of true positives, false positives and false negative respectively for the label y_j.

We apply the Fredman test [20] to verify whether the differences between algorithms are statistical significance. This test is based on method's average rankings cross datasets, and is appropriate for finding significant differences among

Table 2. Performance in terms of 0/1 Loss.

Dataset	BR	CC	MBR	DCC
Music	0.743 (4)	0.713 (3)	0.708 (2)	0.678 (1)
Genbase	0.027 (2.5)	0.027 (2.5)	0.027(2.5)	0.027(2.5)
Medical	0.339 (4)	0.291 (2)	0.327 (3)	0.288 (1)
LangLog	0.757 (3.5)	0.753 (2)	0.757(3.5)	0.736 (1)
Scene	0.491 (4)	0.361 (1)	0.449 (3)	0.389 (2)
Yeast	0.847 (4)	0.779 (1)	0.843 (3)	0.811 (2)
Enron	0.886 (4)	0.881 (2)	0.884 (3)	0.876 (1)
Corel5k	0.995 (4)	0.990 (2)	0.991 (3)	0.988 (1)
Bibtex	0.856 (3.5)	0.847 (2)	0.856(3.5)	0.839 (1)
Ave. Rank	3.722 (4)	1.944 (2)	2.944 (3)	1.389 (1)

different methods. Therefore, for each evaluation measure, we display the values achieved and the rank pertaining to those values for each dataset.

4.3. Threshold Selection

To predict the final output $\hat{y} \in \{0,1\}^L$, we refer to [21] to define a threshold function $f_t(\hat{w})$ such that:

$$\hat{y}_j = \begin{cases} 1 \ if \ \hat{w}_j \geq t \\ 0 \ if \ \hat{w}_j < t \end{cases} \quad (13)$$

where, $\hat{w}_j$ is a confidence values for each label. t_j is a threshold for each $\hat{w}_j$ and is defined in (14):

$$t = \arg\min_t \left\| LCard(D) - (\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{L} 1_{\hat{w}_j} \geq t) \right\| \quad (14)$$

4.4. Experimental Set up

In this experiment, we evaluate all algorithms under a MEKA platform [4, 14] which is a standard open source tool for the evaluation of multi-label algorithms that works on the top of the WEKA framework. Both of the platforms are widely used for research in multi-label classification. SMO (MEKA's implementation of Support Vector Machines based on the SMO algorithm) is used as base classification algorithm. We have considered four classifiers to perform comparisons: binary relevance (BR), classifier chains (CC), meta-BR (MBR), and double layer based classification chains (DCC). The performance of implementations for ensemble binary relevance (EBR), ensemble classifier chains (ECC), and ensemble classifier chains based on double layer (EDCC) are also considered for comparison. Because of the time and memory limitation, we set the number of ensemble iteration m=10 for small and medium datasets, and m=5 for large datasets. The performance of each ensemble method is evaluated using 10-fold cross-validation.

We set 67% of each complete dataset as training sets and the rest of part is used as testing sets. All experiments are run on 64 bit machines.

5. EXPERIMENTS

5.1. Results and Analysis for DCC

(Tables **2-5**) show the performances of DCC's chaining method compared with BR, CC and MBR on 9 benchmark datasets in terms of 0/1 Loss, Hamming Loss, Accuracy and Micro_F1 evaluation measures respectively. The average rankings cross datasets is appropriate for finding significant differences among different methods.

According to (Table **2**), the best average rank (1.389) for 0/1 Loss is achieved by DCC over all types of provided datasets except the performance on Genbase with average ranking 2.5.

The results presented in (Table **3**) show that the MBR model obtains the best results in terms of Hamming Loss and BR takes second place. This is expected, since the BR is actually suitable for most loss functions that ignore label correlations, as demonstrated in [15]. From Table **3**, we see that the performance of DCC with average rank 2.667 is inferior to MBR and BR. However, the differences between the Hamming Loss values of BR and DCC are very small in most of the datasets, except for Music, LangLog and Corel5K. J. Read demonstrated in [4] that it can't be expected that a method performs best over all types of evaluation measures in multi-label classification. Therefore, it comes as no surprise that different methods perform best under different measures.

(Tables **4**, **5)** show that the DCC method also outperforms the other models in terms of Accuracy and Micro_F1 in most of datasets. The average ranks for DCC compared with BR, CC and MBR under Accuracy and Micro_F1 evaluation methods are preponderant.

(Figs. (**5**, **6**)) give the plots of the percentage of training data versus Accuracy and 0/1 Loss. These two figures illustrate how the percentage change of training data affects the enhancement of performance. In this experiment, we take Medical dataset as an example for both comparisons.

(Fig. (**5**)) shows the change of curve for Accuracy under the two pairs of classifiers (for example, BR versus MBR and

Table 3. Performance in terms of Hamming Loss.

Dataset	BR	CC	MBR	DCC
Music	0.197 (1)	0.215 (4)	0.198 (2)	0.209 (3)
Genbase	0.001 (2.5)	0.001 (2.5)	0.001(2.5)	0.001 (2.5)
Medical	0.010 (2.5)	0.010 (2.5)	0.010(2.5)	0.010 (2.5)
LangLog	0.018 (3)	0.018 (3)	0.018 (3)	0.017 (1)
Scene	0.107 (3)	0.106 (2)	0.105 (1)	0.119 (4)
Yeast	0.198 (2)	0.203 (3)	0.197 (1)	0.207 (4)
Enron	0.060 (3.5)	0.060 (3.5)	0.059(1.5)	0.059 (1.5)
Corel5k	0.012 (1)	0.014 (3)	0.013 (2)	0.014 (4)
Bibtex	0.016 (3.5)	0.015 (1.5)	0.016(3.5)	0.015 (1.5)
Ave. Rank	2.444 (2)	2.778 (4)	2.111 (1)	2.667 (3)

Table 4. Performance in terms of Accuracy.

Dataset	BR	CC	MBR	DCC
Music	0.527 (4)	0.546 (3)	0.552 (2)	0.578 (1)
Genbase	0.989 (2.5)	0.989 (2.5)	0.989 (2.5)	0.989(2.5)
Medical	0.751 (4)	0.778 (2)	0.754 (3)	0.781 (1)
LangLog	0.135 (3.5)	0.139 (2)	0.135 (3.5)	0.154 (1)
Scene	0.586 (4)	0.684 (1)	0. 613 (3)	0.650 (2)
Yeast	0.502 (4)	0.538 (1)	0.505 (3)	0.513 (2)
Enron	0.397 (4)	0.399 (2)	0.398 (3)	0.411 (1)
Corel5k	0.085 (4)	0.116 (2)	0.104 (3)	0.124 (1)
Bibtex	0.317 (3)	0.318 (2)	0.316 (4)	0.323 (1)
Ave. Rank	3.667 (4)	1.944 (2)	3.000 (3)	1.389 (1)

Table 5. Performance in terms of Micro_F1.

Dataset	BR	CC	MBR	DCC
Music	0.652 (3.5)	0.652 (3.5)	0.667 (2)	0.677 (1)
Genbase	0.989 (2.5)	0.989 (2.5)	0.989 (2.5)	0.989 (2.5)
Medical	0.805 (3.5)	0.817 (1)	0.805 (3.5)	0.814 (2)
LangLog	0.222 (3.5)	0.227 (2)	0.222 (3.5)	0.242 (1)
Scene	0.679 (3)	0.699 (1)	0.686 (2)	0.662 (4)
Yeast	0.635 (4)	0.655 (1)	0.637 (2)	0.636 (3)
Enron	0.513 (2.5)	0.509 (4)	0.513 (2.5)	0.521 (1)
Corel5k	0.151 (4)	0.176 (2)	0.172 (3)	0.187 (1)
Bibtex	0.404 (3.5)	0.411(2)	0.404 (3.5)	0.415 (1)
Ave. Rank	3.333 (4)	2.111(2)	2.722 (3)	1.833 (1)

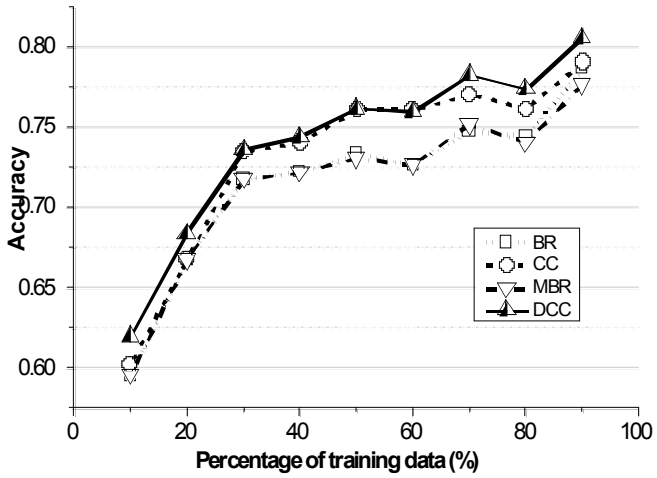

Fig. (5). Accuracy with different percentages of training data.

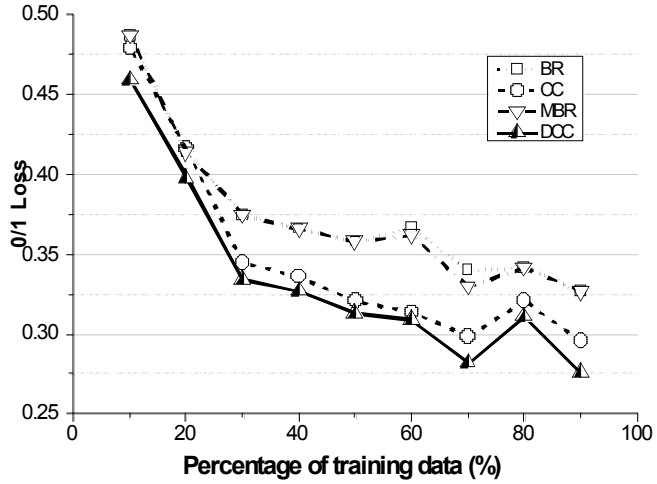

Fig. (6). 0/1 Loss with different percentages of training data.

CC versus DCC) scale with respect to the percentage of training data. From Fig. (**5**), we observe that the Accuracy are improved for the four classifiers when the percentage of training data are increased. For DCC, the improvement of Accuracy displays better than CC when 60% of training data is involved. Moreover, the DCC maintains best all over the four classifiers. But, the degree of improvement for each pair is not too explicit. Especially for BR and MBR, the curves are almost coincident. It means the enhancement of MBR versus MBR is slight.

From Fig. (**6**), we may see the results of comparison in terms of 0/1 Loss. In this figure, as the percentage of training data increasing, all of the results go down and the performances go up. DCC still keeps lowest loss in such a case.

5.2. Results and Analysis for EDCC

(Tables **6-9**) present the results of ensemble implementations for EDCC comparing with EBR, ECC and EMBR on 9 benchmark datasets in terms of in terms of 0/1 Loss, Hamming Loss, Accuracy and Micro_F1 respectively. In comparison with other ensemble classifiers, EDCC yields surprisingly well on all evaluation measures than any other methods, and obtains the highest rankings overall, especially, on large datasets Corel5K and Bibtext.

(Fig. (**7**)) illustrates the influence of different number of iterations on accuracy. Fig. (**7**) shows that when the ensemble number is less than 4, the improvement of prediction accuracy for each classifier has high impact. After that, all the curves keep increase smoothly. For EDCC and ECC, the scope of the improvement under different number of iterations is larger than that on EBR and EMBR. However, the curves for EBR and EMBR show that there is almost no improvement when the iteration numbers are same. In general, the performance of EDCC has a distinct advantage over all the other three classifiers under this circumstance.

The result of Accuracy with different ensemble numbers using four classifiers in terms of 0/1 Loss is shown in Fig. **8**. The figure reveal that when the number of iteration increasing, all of the performances go up. In this point, EDCC still keeps ahead all over the other three classifiers.

Table 6. Performance of ensembles of classifiers in terms of 0/1 Loss.

Dataset	EBR	ECC	EMBR	EDCC
Music	0.728 (3.5)	0.683 (1)	0.728 (3.5)	0.698 (2)
Genbase	0.035 (3)	0.035 (3)	0.035 (3)	0.031 (1)
Medical	0.342 (3.5)	0.321 (2)	0.342 (3.5)	0.306 (1)
LangLog	0.781 (2.5)	0.783 (4)	0.781 (2.5)	0.777 (1)
Scene	0.502 (4)	0.399 (2)	0.438 (3)	0.396 (1)
Yeast	0.855 (4)	0.802 (1)	0.853 (3)	0.824 (2)
Enron	0.874 (3)	0.867 (1)	0.876 (4)	0.869 (2)
Corel5k	0.998 (1.5)	0.999(3.5)	0.998 (1.5)	0.999(3.5)
Bibtex	0.896 (3.5)	0.890 (2)	0.896 (3.5)	0.870 (1)
Ave. Rank	3.167 (4)	2.167 (2)	3.056 (3)	1.611 (1)

Table 7. Performance of ensembles of classifiers in terms of Hamming Loss.

Dataset	EBR	ECC	EMBR	EDCC
Music	0.212 (4)	0.203 (1)	0.210 (3)	0.206 (2)
Genbase	0.001 (2.5)	0.001(2.5)	0.001 (2.5)	0.001 (2.5)
Medical	0.010 (3)	0.010 (3)	0.010 (3)	0.009 (1)
LangLog	0.026 (2.5)	0.026 (2.5)	0.026 (2.5)	0.026 (2.5)
Scene	0.116 (4)	0.101 (1)	0.104 (2.5)	0.104 (2.5)
Yeast	0.208 (3.5)	0.205 (1)	0.208 (3.5)	0.207 (2)
Enron	0.057 (3.5)	0.056 (1.5)	0.057 (3.5)	0.056 (1.5)
Corel5k	0.024 (3.5)	0.025 (1.5)	0.026 (3.5)	0.025 (1.5)
Bibtex	0.022 (3.5)	0.021 (2)	0.022 (3.5)	0.020 (1)
Ave. Rank	3.333(4)	1.778 (1)	3.056 (3)	1.833 (2)

Table 8. Performance of ensembles of classifiers in terms of Accuracy.

Dataset	EBR	ECC	EMBR	EDCC
Music	0.557 (4)	0.567 (3)	0.566 (2)	0.569 (1)
Genbase	0.986 (3)	0.986 (3)	0.986 (3)	0.988 (1)
Medical	0.767 (3)	0.785 (2)	0.766 (4)	0.794 (1)
LangLog	0.166 (2.5)	0.165 (4)	0.166(2.5)	0.167 (1)
Scene	0.627 (4)	0.704 (1)	0.654 (3)	0.678 (2)
Yeast	0.533 (4)	0.543 (1)	0.534 (3)	0.538 (2)
Enron	0.441 (4)	0.454 (1)	0.442 (3)	0.451 (2)
Corel5k	0.122 (4)	0.126 (3)	0.129 (2)	0.132 (1)
Bibtex	0.310 (2.5)	0.318 (2)	0.310(2.5)	0.329 (1)
Ave. Rank	3.444 (4)	2.222 (2)	2.778 (3)	1.333 (1)

Table 9. Performance of ensembles of classifiers in terms of Miro_F1.

Dataset	EBR	ECC	EMBR	EDCC
Music	0.672 (4)	0.674(2.5)	0.678 (1)	0.674 (2.5)
Genbase	0.985 (3)	0.985(3)	0.985 (3)	0.987 (1)
Medical	0.817 (3)	0.826 (2)	0.816 (4)	0.831 (1)
LangLog	0.255 (3)	0.255 (3)	0.255(3)	0.255(1)
Scene	0.694 (4)	0.731 (1)	0.709 (3)	0.712 (2)
Yeast	0.660 (3.5)	0.662 (1)	0.661 (2)	0.660 (3.5)
Enron	0.558 (4)	0.564 (1)	0.560 (3)	0.562 (2)
Corel5k	0.194 (4)	0.200(2.5)	0.202(2.5)	0.212 (1)
Bibtex	0.391 (3.5)	0.403 (2)	0.391(3.5)	0.413 (1)
Ave. Rank	3.556 (4)	2.000 (32)	2.778 (3)	1.667 (1)

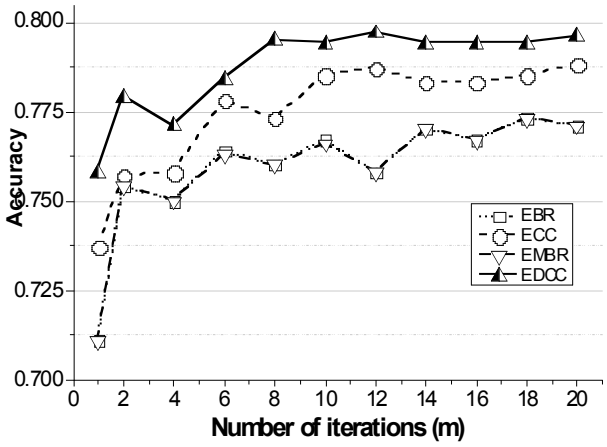

Fig. (7). Accuracy with different ensemble numbers under four classifiers in terms of Accuracy.

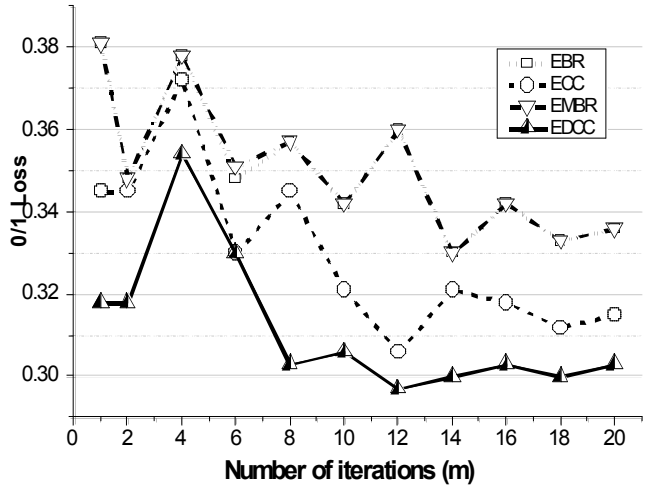

Fig. (8). Accuracy with different ensemble numbers under four classifiers in terms of 0/1 Loss.

CONCLUSION

This paper presents a chaining method for multi-label classification based on double layers. We derive this algorithm from the binary relevance method, which many literatures argued has many advantages over more sophisticated methods currently. The algorithm sets two layers to decompose the multi-label classification problem into independent binary classification problems respectively. The instance with attribute space for each binary model is extended with label relevance of all previous classifiers to form a classifier chain. The label correlation information is passed along a chain of classifiers to help learning and predicting. This method counteracts the disadvantages of the binary relevance method and inherits the benefit of classifier chain. The ensembles of classifiers chains are also implemented to augment predictive performance.

By using a considerable variety of multi-label datasets and evaluation measures, we carried out empirical evaluations compared with a range of algorithms. The experiments results demonstrate that DCC and EDCC algorithms can achieve better predictive performance in most of datasets which is used in this experiments and present superior to related methods.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by National Science and Technology Planning Project of China (2012BAH76F01).

REFERENCES

[1] A. Elisseeff and J. Weston, "*A kernel method for multi-labelled classification*", In: Advances in Neural Information Processing Systems 14, MIT Press, Cambridge, MA, pp. 681-687, 2002.

[2] G. Irie, T. Satou, A. Kojima, T. Yamasaki, and K. Aizawa, "Affective audio-visual words and latent topic driving model for realizing movie affective scene classification", *IEEE Transactions on Multimedia*, vol. 12, no.6, pp. 523-535, 2010.

[3] G. Tsoumakas, M. Zhang, Z. Zhou, "Introduction to the special issue on learning from multi-label data", *Machine Learning*, vol. 88, pp. 1-4, 2012.

[4] J. Read, B. Pfahringer, G. Holmes, "Generating synthetic multi-label data streams", *In: MLD'09: 1st ECML/PKDD 2009 Workshop on Learning from Multi-Label Data*, 2009.

[5] A. Dimou, G. Tsoumakas, V. Mezaris, I. Kompatsiaris, I. Vlahavas, "An empirical study of multi-label learning methods for video annotation", *In: Proceedings of the 7th International Workshop on Content-Based Multimedia Indexing*", Chania, Crete, Greece, pp.19-24, 2009.

[6] C. Vens, J. Struyf, L. Schietgat, S. D˘zeroski, H. Blockeel, "Decision trees for hierarchical multi-label classification", *Machine Learning*, vol. 2, no. 73, pp.185-214, 2008.

[7] G. Tsoumakas, I. Katakis, and I. Vlahavas, "*Mining Multi-Label Data*," In: Data Mining and Knowledge Discovery Handbook, 2nd ed., Springer, pp. 667-685, 2010.

[8] G. Tsoumakas, I. P. Vlahavas, Random k-labelsets, "An ensemble method for multi-label classification". *In: ECML '07: 18th European Conference on Machine Learning*, Warsaw, Poland, pp. 406-417, Springer 2007.

[9] M. Boutell, J. Luo, X. Shen, and C. Brown, "Learning multi-label scene classification", *Pattern Recognition*, vol. 37, no. 9, pp.1757-1771, 2004.

[10] K. Trohidis, G. Tsoumakas, G. Kalliris, and I. P. Vlahavas, "Multi-label classification of music by emotion", *ISMIR*, pp. 325-330, 2008.

[11] M. Zhang, Z. Zhou, "Exploiting unlabeled data to enhance ensemble diversity", *Data Mining and Knowledge Discovery,* vol.26, no.1, pp. 98-129, 2013.

[12] S. Li, N. Li, "Multi-label Data Mining: A Survey", *Computer Science*, vol.40, no.4, pp. 14-21, 2013.

[13] S. Godbole, S. Sarawagi, "Discriminative methods for multi-labeled classification", In: PAKDD '04: 8th *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Australia, Springer, pp. 2-30, 2004.

[14] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification", *Machine Learning*, vol. 85, no. 3, pp. 333-359, 2011.

[15] E. C. Gonçalves, A. Plastino, "A genetic algorithm for optimizing the label ordering in multi-label classifier chains", *IEEE Congress on Evolutionary Computation*, pp. 454-461, 2013.

[16] G. Madjarov, D. Kocev, D. Gjorgjevikj, and D. Saso, "An extensive experimental comparison of methods for multi-label learning", *Pattern Recognition*, vol. 45, 2012.

[17] M. L. Zhang, and Z. H. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, no. 7, 2007.

[18] G. Tsoumakas, E. Spyromitros, J. Vilcek, and I. Vlahavas, "Mulan: A java library for multi-label learning," *Journal of Machine Learning Research*, vol.12, pp. 2411-2414, 2011.

[19] M. Boutell, J. Luo, X. Shen, and C. Brown, "Learning multi-label scene classification", *Pattern Recognition*, vol.37, no. 9, pp.1757-

1771, 2004.

[20] J. Demsar, "Statistical comparisons of classifiers over multiple data sets", *J machine learn res* vol.7, pp. 1-30, 2006.

[21] R. E. Fan, C. J. Lin, "A study on threshold selection for multi-label classification. Tech. rep", National Taiwan University, 2007. http://www.cise.ntu.edu.tw/~cjlin/papers/threshold.pdf.

Received: September 22, 2014 Revised: November 30, 2014 Accepted: December 02, 2014